REMARKS/ARGUMENTS

Claims 11-15 have been presented in a form that is believed to be preferred by the Examiner. Other claims have been amended to further clarify the subject matter regarded as the invention. It is respectfully requested that all rejections under 35 U.S.C. §112 be withdrawn. It should be noted that no substantive claim amendment has been made; however, new claims 29 and 30 have been amended. Thus, claims 1-30 are now pending.

In the Office Action, the Examiner rejected claims 1-9 and 11-21 under 35 U.S.C. §102 over U.S. patent No. 6,292,883 B1 (*Augusteijn et al.)* This rejection is fully traversed below.

The application relates to loading and executing portable, platform independent programming instructions within a virtual machine. One aspect of the present invention provides a mechanism that will generally improve the runtime performance of virtual machines by eliminating the need to always process data (e.g., traverse a constant pool) at runtime to execute virtual machine instructions. In effect, the described system contemplates doing some extra work during the load time and representing that information in a form that can be used more efficiently at runtime. (See, for example, abstract.)

Other aspects of the invention provide techniques for creating data structures that are suitable for use within a virtual machine. By way of example, in one embodiment, an enhanced Java™ bytecode representation having a pair of Java™ compliant bytecode streams is disclosed. The enhanced Java™ bytecode has a Java™ code stream suitable for storing various Java™ commands as bytecodes within a code stream. A Java™ data stream is used to store the data parameters associated with the Java™ commands in the code stream. As will be appreciated, the invention allows for representation of actual parameter values, or references to actual parameter values, in the code stream. Accordingly, data parameters can be provided for efficient execution of Java™ instructions without requiring further processing of Constant Pools at run time. As a result, the performance of Java™ complaint virtual machine can be enhanced

using a pair of streams in the context of the invention. (See, for example, Specification, Page 5, lines 4-24.)

Claim 1 pertains to a method for creating data structures suitable for use by a virtual machine. The virtual machine can use the data structures to execute computer instructions. Claim 1 recites converting a stream of commands and data associated with the commands into a pair of streams for use in the virtual machine. It should be noted that the pair of streams include a code stream and a data stream, wherein the code stream includes the commands and the data stream includes the data associated with the commands in the code stream.

In the Office Action, the Examiner has asserted that *Augusteijn et al.* teaches converting a stream of commands and data associated with the commands into a pair of streams in the context of the invention. (Office Action, page 4, paragraph 7). The Applicant respectfully disagrees.

Augusteijn et al. pertains to converting virtual machine instructions into native instructions. (*Augusteijn et al.*, Abstract). As such, *Augusteijn et al.* illustrates a microcontroller 110 comprising a processor 112 with a pre-determined microcontroller core 114, referred to as a native machine, for executing native instructions. In addition, a pre-processor 130 is provided to convert a virtual machine instruction, fetched from a memory 120, into at least one native instruction. (*Augusteijn et al.*, Fig. 1, Col. 7, 3-30).

Contrary to the Examiner's assertion, it is respectfully submitted, converting a virtual machine instruction into one or more native instructions does not teach converting a virtual machine instruction in a first stream into a pair of streams for use in the virtual machine. Again, it should be noted that the pair of streams include a code stream and a data stream, wherein the code stream includes the commands and the data stream includes the data associated with the commands in the code stream. Accordingly, it is respectfully submitted the rejection of claims 1, 11 and 21 is improper and should be withdrawn.

Moreover, it is respectfully submitted that *Augusteijn et al.* does not teach or suggest converting a virtual machine instruction into a pair of streams in the context of the invention. This is evident because, among other things, the methodology taught by *Augusteijn et al.* pertains to defining a program-specific virtual machine for program statements of a source program. The virtual machine provides a set of virtual machine

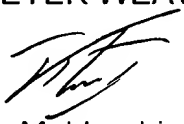
instructions that typically take less memory to store than native instructions. (*Augusteijn et al.,* abstract). *Augusteijn et al.,* however, does not teach or suggest converting a virtual machine instruction itself into a pair of streams (e.g., a code and a data stream). Thus, claims 11, 11 and 21 are patentable over *Augusteijn et al.* for at least this reason alone. In addition, claims that are dependent on claims 1, 11 and 21 are patentable at least for this reason. Moreover, these dependent claims recite additional features which render them patentable for additional reasons. For example, claim 4 recites writing a representation of a first command associated with a first instruction into a code entry of the code stream, determining whether the first command has data associated with it, and writing a representation of the associated data or a reference to a representation of the data associated with the first command into a first data entry of the data stream when the command has associated data. Contrary to the Examiner's assertion, it is respectfully submitted that *Augusteijn et al.*, does not teach these features.

Claim 16 pertains to a method for executing computer instructions on a virtual machine. As such, claim 16 among other things, recites fetching a command associated with a virtual machine computer instruction from a code stream, and fetching from a data stream the associated parameter of the command when it is determined that the command has an associated parameter. As noted above, *Augusteijn et al.* does not teach or suggest providing a code stream and a data stream for a virtual machine in the context of the invention. Hence, *Augusteijn et al.* cannot possibly teach the features. Accordingly, it is respectfully submitted that claim 16 and its dependent claims are patentable over *Augusteijn et al.* for at least these reasons. For example, it should be noted that claim 22, among other things, recites precrossing data from a constant pool and writing a representation (or a reference) of the data in the code stream. It is respectfully submitted that these features are not taught or suggested by *Augusteijn et al.*

Based on the foregoing, it is submitted that claims 1-28 are patentably distinct over the cited art of record. Additional limitations recited in the independent claims or the dependent claims are not further discussed because the limitations discussed above are sufficient to distinguish the claimed invention from the cited art. Accordingly, Applicant believes that all pending claims are allowable and respectfully requests a Notice of Allowance for this application from the Examiner.

Applicants hereby petition for an extension of time which may be required to maintain the pendency of this case, and any required fee for such extension or any further fee required in connection with the filing of this Amendment is to be charged to Deposit Account No. 500388 (Order No. SUN1P814). Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP



R. Mahboubian
Reg. No. 44,890

P.O. Box 778
Berkeley, CA 94704-0778